Designing KDD-Workflows via HTN-Planning for Intelligent Discovery Assistance

Jörg-Uwe Kietz[1] and **Floarea Serban**[1] and **Abraham Bernstein**[1] and **Simon Fischer**[2]

Abstract. Knowledge Discovery in Databases (KDD) has evolved a lot during the last years and reached a mature stage offering plenty of operators to solve complex data analysis tasks. However, the user support for building workflows has not progressed accordingly. The large number of operators currently available in KDD systems makes it difficult for users to successfully analyze data. In addition, the correctness of workflows is not checked before execution. Hence, the execution of a workflow frequently stops with an error after several hours of runtime.

This paper presents our tools, eProPlan and eIDA, which solve the above problems by supporting the whole life-cycle of (semi-) automatic workflow generation. Our modeling tool eProPlan allows to describe operators and build a task/method decomposition grammar to specify the desired workflows. Additionally, our Intelligent Discovery Assistant, eIDA, allows to place workflows into data mining (DM) tools or workflow engines for execution.

1 Introduction

One of the challenges of Knowledge Discovery in Databases (KDD) is assisting users in creating and executing KDD workflows. Existing KDD systems such as the commercial IBM SPSS Modeler[3] or the open-source KNIME[4] and RapidMiner[5] support the user with nice graphical user interfaces. Operators can be dropped as nodes onto the working pane and the data-flow is specified by connecting the operator-nodes. This works very well as long as neither the workflow becomes too complicated nor the number of operators becomes too large.

However, in the past decade, the *number of operators* in such systems has been growing fast. All of them contain over 100 operators and RapidMiner, which includes Weka, R, and several pluggable operator sets (such as anomaly detection, recommendation, text and image mining) now has around 1000. It can be expected that the transition from closed systems (with a fixed set of operators) to open systems that can also use Web services as operators (which is especially interesting for domain specific data access and transformations) will further accelerate the rate of growth *resulting in total confusion about what operators to use for most users*.

In addition to the number of operators also the *size of the KDD workflows* is growing. Today's workflows easily contain hundreds of operators. Parts of the workflows are applied several times (e.g. the preprocessing sub-workflow has to be applied on training, testing, and application data) implying that the users either need to copy/paste or even repeatedly design the same sub-workflow[6] several times. As none of the systems maintain this "copy"-relationship, it is left to the user to maintain the relationship in the light of changes.

Another weak point is that workflows are not checked for *correctness* before execution. As a consequence, the execution of the workflow oftentimes stops with an error after several hours runtime due to small syntactic incompatibilities between an operator and the data it should be applied on.

To address these problems several authors [1, 3, 18] propose the use of planning techniques to automatically build such workflows. However, all these approaches are limited. First, they only model a very small number of operations and were only demonstrated to work on very short workflows (less than 10 operators). Second, none of them models operations that work on individual columns of a data set: they only model operations that process all columns of a data set in the same way. Lastly, the approaches cannot scale to large amounts of operators and large workflows: their planning approaches fail in the large design space of "correct" (but nevertheless most often unwanted) solutions. A full literature review about IDAs (including these approaches) can be found in our survey [13].

In this paper we describe the first approach for designing KDD workflows based on ontologies and Hierarchical Task Network (HTN) planning [5]. *Hierarchical task decomposition knowledge* available in DM (e.g. CRISP-DM [2] and CITRUS [15]) can be used to significantly reduce the number of generated unwanted correct workflows. The main scientific contributions of this paper, hence, are: First, we show how KDD workflows can be designed using ontologies and HTN-planning in eProPlan. Second, we exhibit the possibility to plug in our approach in existing DM-tools (as illustrated by RapidMiner and Taverna). Third, we present an evaluation of our approach that shows significant improvement and simplification of the KDD-workflow design process. Thus, the KDD researchers can easily model not only their DM and preprocessing operators but also their DM tasks that is exploited to guide the workflow generation. Moreover less experienced users can use our RM-IDA plugin to automatically generate workflows in only 7-clicks. Last but not least, the planning community may find it interesting to see this real world problem powered by planning techniques and may also find some of the problems we faced and solved rather pragmatically inspiring for

[1] University of Zurich, Department of Informatics, Dynamic and Distributed Information Systems Group, Binzmühlestrasse 14, CH-8050 Zurich, Switzerland {kietz|serban|bernstein}@ifi.uzh.ch
[2] Rapid-I GmbH, Stockumer Str. 475, 44227 Dortmund, Germany fischer@rapid-i.com
[3] http://www.ibm.com/software/analytics/spss/
[4] http://www.knime.org/
[5] http://rapid-i.com/content/view/181/190/

[6] Several operators must be exchanged and cannot be reapplied. Consider for example training data (with labels) and application data (without labels). Label-directed operations like feature-selection cannot be reapplied. But even if there is a label on separate test data, redoing feature selection may result in selecting different features. To apply and test the model, however, exactly the same features have to be selected.

further research.

The rest of this paper is structured as follows: Section 2 describes the knowledge used for planning, then Section 3 details the components of our system. Section 4 describes several evaluation methods and, finally, we conclude with Section 5.

2 The Planning Knowledge

We modeled our Data Mining Workflow planning problem as a Data Mining Worlkflow ontology (DMWF),[7] which we succinctly illustrate here (a more detailed description of it can be found in [8, 9]).

The DMWF contains *input/output objects* and their *meta-data* (Sec.2.1), which are sufficiently detailed to enable column-wise operations – a feature that is not available in any of the previous approaches. In addition, *tasks and methods* are used to guide and simplify the planning process. Each method consists of a sequence of steps, where each step is a (sub-) task or an operator (Sec. 2.2 shows some of the methods). In total, the DMWF contains *more than 100 operators* from RapidMiner (Sec. 2.3 illustrates one). The amount of operators and their partial redundancy made it favorable to structure them in an inheritance hierarchy starting with abstract operators until the basic operators which can be applied on the data. The number of operators is not a limitation of the HTN-planning approach, but a limitation set by the effort to model them and to keep them consistent with changes introduced by new releases of RapidMiner. To get a significantly higher number of modeled operators, semi-automatic modeling or at least verification methods need to be developed.

Besides the main contribution of supporting users in designing KDD workflows, this paper may also be interesting to the planning community because it shows the successful usage of planning techniques to solve the problem of workflow design and more generally problem-specific software-configuration. It may stipulate further research on planning as we solved some problems that did not get much attention in planning so far. First, in our domain it is usually easy to find a correct plan. The simplest correct plan for prediction uses the default model (mean-value for regression, modal-value for classification). This is a correct solution for all predictive modeling problems, but it is only the baseline that DM wants to improve and not the wanted solution. We tackle that problem by excluding such unwanted workflows from our HTN. The real problem is not finding a solution, but handling the large amount of solutions[8]. We handle this by grouping of plans based on meta-level equivalent output (similar characteristics of the output) [9] and by using the probabilistic pattern generated by meta-learning (see Sec. 2.4) not only to rank the enumerated workflows, but also for a heuristic beam search in the space of possible solutions. Another interesting problem is the large number of relevant operators that we handled by embedding conditions and effects into an ontological operator hierarchy with inheritance. This is supported by our eProPlan plugin into the popular ontology editor Protégé. Furthermore, RapidMiner (and in DM in general) has several special purpose control/loop operators like cross-validation. They are parametrized operators (the number of folds and the sampling strategy for cross validation). In contrast to other operators, it is not sufficient to choose the dominating operators since they contain one or more subtasks that have to be planned as well. This is similar to planning general control structures like if-then-else or loops, but the problem is also easier to solve as these dominating operators and their subtasks have a special purpose and, therefore, task/method decompositions as all other tasks. Figure 1a shows a cross-validation workflow which uses first a preprocessing task and then it applies cross-validation to produce and test the model as seen in Figure 1b. During the training step it selects the important features and then it trains the model. The produced model is then applied on the testing data. The output is an average of the n runs (where n is the number of folds, usually set to 10).

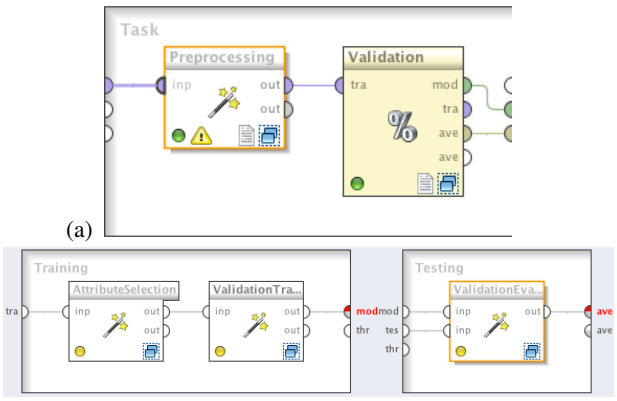


Figure 1: (a) Cross Validation as Operator (labeled as "Validation" in the Figure) in a workflow; (b) Subtasks of Cross Valdation

2.1 Meta-Data to describe Input/Output Objects

The planner recognizes the *IO-Objects* (e.g. *DataTable*, *DocumentCollection* and *ImageCollection*), *Background Knowledge*, *Model* (e.g. *PreprocessingModel* and *PredictionModel*, recording required, modified, added and deleted attributes such that the conditions and effects of applying these models on the data can be computed by the planner), and *Report* (e.g. *PerformanceVector* and *LiftChart*). As an example Table 1 shows the meta-data of a *DataTable*. The meta-data for the user-data is generated by RapidMiner's/RapidAnalytic's meta-data analyzer and passed to the planner. During the planning process the planner generates the meta-data of an operator's output objects from the operator's effect-specification (see Sec. 2.3).

Attribute	#Attr	Type	Role	#Diff	#Miss	Values	Min	Max	Mean	Modal	Std.
age	1	Scalar	input		0	[]	20.0	79.0	50.95		16.74
genLoad	1	Nominal	input	2	36	[0,1]				1	
label	1	Nominal	target	2	0	[+,-]				+	
meas1	1	Scalar	input		0	[]	0.10	3.93	1.845		0.861
meas2	1	Scalar	input		30	[]	0.12	4.35	1.979		0.902
meas3	1	Scalar	input		0	[]	0.33	5.35	2.319		1.056
sex	1	Nominal	input	2	0	[f,m]				m	

Table 1: Meta-Data for a Data Table

One of the strengths of the IDA is the ability to plan workflows with attribute-wise operations – a feature no other previous approach had so far. Especially biological micro-array data can easily contain several thousand columns, turning this possibility into a major performance bottleneck. Looking at a number of such analyses we observed that these columns often have very similar (if not identical)

[7] It is public available from `http://www.e-LICO.eu/ontologies/dmo/e-Lico-eProPlan-DMWF-HTN.owl`. The best way to open this ontology is: download Protégé 4.0 or 4.1 from `http://protege.stanford.edu/` and eProPlan from `http://elico.rapid-i.com/eproplan.html` (2.0.1 for Protégé4.0 and 2.1.0 for 4.1).

[8] With column-wise operations this may be very large, just consider having 5 alternative methods to discretize 100 attributes. This results in $5^{100} \approx 10^{70}$ possible correct plans.

[9] 'Meta-level equivalent output' can be defined as the IOOs-descriptions produced by the planner are equivalent up to the names of individuals.

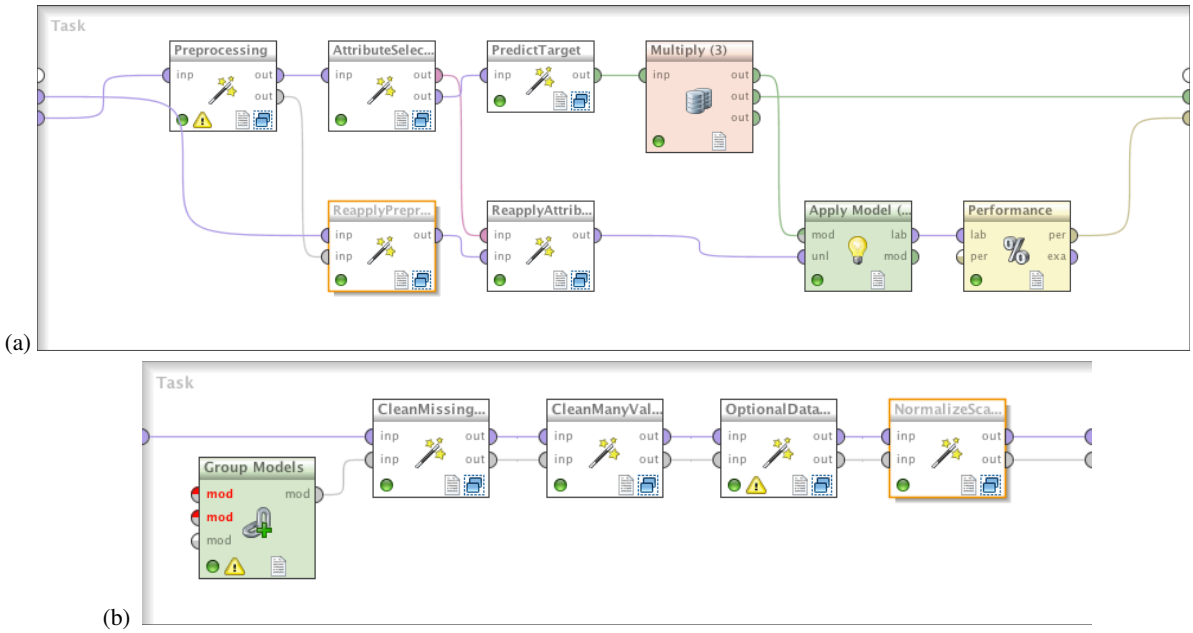


Figure 2: Methods for (a) Modeling with Test-Set Evaluation and (b) Preprocessing

meta-data and data characteristics, effectively eliminating the need to differentiate between them during planning. To keep the strength and avoid the performance bottleneck of several thousand attributes that are identical on the meta-data level, we introduced attribute groups that collect all attributes together where the operator conditions would not discriminate between them anyway. This worked well as all column-wise operators were able to handle not only single columns but also specified sets of columns. Therefore, we are able to generate workflows with attribute-group-wise operations for microarray and collaborative filtering data with even 50.000 attributes.

There are various approaches which have used meta-data to suggest the best algorithm for a given problem [12, 4, 7] in the context of meta-learning.

2.2 The Task/Method decomposition

The top-level task of the HTN is the *DM* task. It has six methods: *Clustering*, *Association Rule Mining*, *Predictive Modeling with Test Set Evaluation* (external given separation), *Predictive Modeling with Cross Validation*, *Predictive Modeling with Test Set Split Evaluation* (random 70:30 split), and *Simple Modeling with Training Set Performance*.

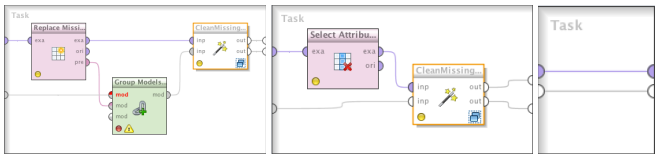
The selection is directed by the (required) main-goal, i.e. *Pattern Discovery* enforces *Association Rule Mining*, *Descriptive Modeling* enforces *Clustering*, and *Predictive Modeling* forces the choice of one of the others. If test data are provided *Modeling with Test Set Evaluation* has to be chosen, otherwise the choice can be influenced by an (optional) evaluation-subgoal (not possible with the current GUI). If there are still several methods possible, they are enumerated in the rank-order provided by the probabilistic planner (see Sec. 2.4). Each method consists of a sequence of steps, where each step is a (sub-)task or an operator (it can also be an abstract operator subsuming several basic or dominating operators). Planning occurs in the order of steps, however the resulting data flow is not necessarily linear, as the grammar allows the specification of port mapping. Figure 2a shows the method *Modeling with Test Set Evaluation* and its flow of IO-Objects in RapidMiner.[10] The white nodes are tasks to be planned and the other nodes are operators. Operators in RapidMiner are grouped in a way similar to the DMWF. Some of the top nodes have colors. The greenish ones are operators that deal with *Model* as well as objects that inherit from *Model*. This includes all learners and the *Apply Model* operator. The more purple ones are data transformation operators. The RapidMiner's plan-interpreter adds a Multiply node whenever an IO-Object is used by several operators.

The *Preprocessing* task has only one method (Fig. 2b). First an empty preprocessing model is created using the operator "Group Models". It is then extended by the next steps. All its sub-tasks have optional "Nothing to Do" methods (as shown for *CleanMissingValues* in Fig. 3c). Most of the preprocessing methods are recursive, handling one attribute at a time until nothing is left to be done. *CleanMissingValues* has two different recursive methods, the choice is made by the planner depending on the amount of missing values. If there are more than 30% values missing, the attribute can be dropped (Figure 3b). When there are less than 50% missing, it can be filled with mean or modal value (Figure 3a). If $30 - 50\%$ of the values are missing, both methods can be used and plans for both are enumerated in the order provided by the probabilistic planner. The usage of column-wise operations is illustrated in Figure. 3, which shows how depending on the amount of missing values per column the planner chooses to fill or drop the column.

Figure 4 shows a generated workflow for the UCI data set labor-negotiations, which has 16 attributes with various amounts of missing values. Note that in its output the planner compresses recursive tasks into a single task to simplify the browsing of the workflow by the user. Such column-wise handling can greatly improve the results of workflows. For users, however, it is usually too much manual effort (placing and connecting the 22 operations and setting their parameters in the workflow below).

In total the HTN of the DMWF now contains 16 tasks with 33

[10] Note that the figures illustrate the methods with structurally equivalent workflows in RapidMiner and do not show the more complicated method definitions in the ontology.



(a) (b) (c)

Figure 3: The (a) “Fill Missing Values”, (b) “Drop Missing Values”, and (c) “Empty”/“Nothing to Do” Method that can be used in Fig. 2b

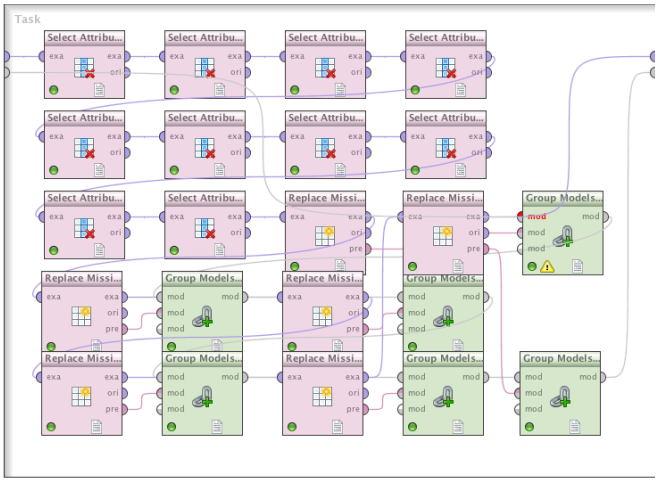


Figure 4: Resulting Example Workflow for Missing Value Cleaning

methods, such that it handles general DM from single table data sets with adequate preprocessing. However, we believe that this will show its real power in customer/application area specific grammars designed and/or extended in eProPlan, resulting in adaptive customer templates for workflows. The complexity of the produced workflows depends on the characteristics of the dataset (if it has missing values, if it needs to be normalized, etc.) and on the number of modeled operators for each steps of the KDD process (*FeatureSelection* and *DataMining* have more operators). Due to space limitations, we only illustrate some parts of the grammar here. The full grammar can be inspected in the public available DMWF-Ontology[11].

2.3 The Operator Models

To be able to express the operators' conditions and effects for planning we stored them as annotations in the ontology. Conditions and effects can contain concept expressions, SWRL-rules,[12] and some extended-SWRL-like built-ins. We have introduced a set of special *built-ins* needed for planning (e.g., *new*, *copy*, *copyComplex*, etc.). These built-ins allow to create, copy, and destroy objects during planning (e.g., models, produced IO-objects, weights, etc.). eProPlan allows to define new built-ins which are stored as subclasses of the *Built-in* concept. Each built-in can have types/parameters and the corresponding implementation in Flora2 [16]. But users who want to add new built-ins need to have some Flora-2 knowledge. They have the possibility to define new functions/operations on the data and introduce them in the conditions and effects. The built-ins' definition with parameters and implementation are stored as class annotations.

[11] Use an OWL2 Ontology Editor like Protege to ”Open OWL ontology from URI” with `http://www.e-lico.eu/ontologies/dmo/e-Lico-eProPlan-DMWF-HTN.owl`

[12] `http://www.w3.org/Submission/SWRL/`

Inputs and outputs of an operator are defined as concept expressions and are either stored as superclasses or as equivalent classes. The parameters and the corresponding RapidMiner operator name are stored in equivalent classes. Fig. 5 exemplifies the abstract operator for a classification learner operator with its corresponding inputs/outputs, condition and effect.

”ClassificationLearner”:
Equiv. class: *PredictiveSupervisedLearners* **and**
(*uses* **exactly 1** *DataTable*) **and**
(*produces* **exactly 1** *PredictionModel*) **and**
(*operatorName* **max 1** Literal)
Condition: [*DataTable* **and** (*targetAttribute* **exactly 1** *Attribute*) **and**
(*inputAttribute* **min 1** *Attribute*) **and**
(*targetColumn* **only** (*DataColumn* **and**
columnsHasType **only** (*Categorial*))) **and**
(*inputColumn* **only** (*DataColumn* **and**
columnsHasType **only** (*Scalar* **or** *Categorial*)))
](?D)
→ *new*(**?this**), *ClassificationLearner*(**?this**), *uses*(**?this**,?D)
Effect: *uses*(**?this**,?D), *ClassificationLearner*(**?this**),
inputColumn(?D,?IC),*targetColumn*(?D,?TC),
→ *copy*(?M,?D, {*DataTable(?D)*, *containsColumn*(?D,?_),
amountOfRows(?D,?_)}),*produces*(**?this**,?M), *PredictionModel*(?M),
needsColumn(?M,?IC), *predictsColumn*(?M,?TC)

Figure 5: An abstract operator: ClassificationLearner

A basic classification learner operator inherits all the characteristics of the classification learner. In addition, it can define more refined conditions or effects and more parameters. Fig. 6 shows the refinement of the general class of all classification learners to a specific Support Vector Machine implementation in RapidMiner. It has an additional condition (binary target and scalar input attribute and no attribute is allowed to have missing values), but it does not contain a refined effect. Its effect is the one used for all classification learners (it builds a predictive model that requires all input attributes to be present to be applicable and predicts the target attribute).

”RM_Support_Vector_Machine_LibSVM_C_SVC_linear”:
Equiv. class: RM_Operator **and**
(*usesData* **exactly 1** *DataTable*) **and**
(*producesPredictionModel* **exactly 1** *PredictionModel*) **and**
(*simpleParameter_kernel_type* **value** ”linear”) **and**
(*simpleParameter_svm_type* **value** ”minimal_leaf_size”) **and**
(*operatorName* **exactly 1** {”support_vector_machine_libsvm”})
Condition: [*MissingValueFreeDataTable* **and**
(*targetColumn* **exactly 1** *CategorialColumn*) **and**
(*inputColumn* **min 1** *Thing*) **and**
(*inputColumn* **only** (*ScalarColumn*))
](?D)
→ *RM_Support_Vector_Machine_LibSVM_C_SVC_linear*(**?this**),
simpleParameter_svm_type(**?this**,”C-SVC”),
simpleParameter_kernel_type(**?this**,”linear”)

Figure 6: A basic classification learner operator

Each input/output class expression (e.g., *usesData exactly 1 DataTable*) has an annotation which defines its port mapping to its corresponding RapidMiner operator port (e.g., ”training set”). Both conditions and effects are rules. Conditions check the applicability (lhs) and infer the parameter settings (rhs); different solutions can

infer that the operator can be applied with different parameter settings. Effects compute the variable-bindings (lhs) for the assertion to be made (rhs); all different solutions are asserted as the effect of one operator application.

2.4 Probabilistic Ranking of Workflows

The HTN-planning method described in this paper enumerates all possible workflows for a given problem. The operator models ensure that they are executable without error. Furthermore, the HTN-Grammar prevents senseless operator combinations. For example, first normalizing the data and then discretizing it does not make sense since the normalization effect is absorbed by the discretization one. Also, converting the scalar data to nominal and then converting it back is a useless operation. Another example is dropping attributes without a reason. Nonetheless, the planner can still generate a very large number of correct candidate workflows and we are unaware of any analytical knowledge available to decide which of them will perform well on the current data. Meta-learning tries to learn relations between data characteristics and method performance. Hence, the IDA uses such meta-learned [11] patterns to order the enumeration of M candidate workflows (heuristic search) and to finally select the N best plans and present them to the user. The ranking approach works as follows: whenever several operators ('meta-level NON equivalent output') or methods are applicable, the PP is asked for a (local, up-to now) ranking and delivers the plans in this order. This determines which space is enumerated if the planner is asked for a limited number of solutions. In the end, all generated alternatives are ranked (with left and right context available) for the final presentation to the user.

The planning knowledge described above also does not know about the execution time of operators. This is caused by the fact that the actual runtime of a DM method cannot be predicted easily because of the complexity of the generated models. It can be worst-case bounded in the number of examples and attributes, but its actual size is strongly affected by statistical properties (e.g. noise-level) of the data. The runtime prediction of a DM method was first introduced in [6]. The ranking in our planner, therefore, relies on a meta-learning based method to predict the runtime of modeling and feature selection operators [19].

3 The Overall System

Our system has two main components as illustrated in Fig. 7: eProPlan, our modeling support tool for new operators and new tasks to be solved by the planner, and eIDA, which generates and deploys workflows into DM-suites. eProPlan is the modeling environment for the DM Workflow Ontology (DMWF). It allows to model new operators and uses a task-method decomposition grammar to solve DM problems. Designed as a plugin for the open-source ontology-editor Protégé 4,[13] eProPlan exploits the advantages of the ontology as a formal model for the domain knowledge. Instead of employing the ontological inferences for planning (as done in [3, 17]) we extend the ontological formalism with the main components of a plan, namely operator conditions and effects for classical planning and tasks-methods decomposition grammar for HTN-planning. This allowed us to cleanly separate the inheritance from the planing mechanisms in our systems.

[13] http://protege.stanford.edu/

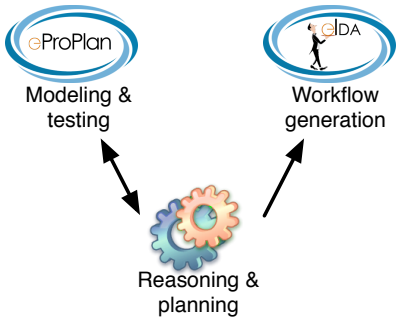


Figure 7: The eProPlan architecture

The planner is implemented in Flora2/XSB [16] and uses different parts from the workflow ontology for different purposes (see Figure 8). Specifically, it employs both a specialized ABox (assertional box—individual assertions) reasoner that relies on an external TBox (terminological box: classes and properties) reasoner (e.g. Pellet[14] or FaCT++ [14]) as a subroutine. Since the TBox reasoner can only partially handle OWL2 (Web Ontology Language[15]), we filter all expressions that are not supported from the ontology. The resulting inferred/completed TBox and its possibly inconsistent class definitions are passed to our ABox reasoner. The ABox reasoner, implemented in Flora2/XSB, first compiles the classified TBox obtained from Pellet on the initial ontology. Then, we process the operators together with their inputs, outputs, preconditions, and effects that are stored as OWL annotations. Tasks and methods are handled analogously. Finally, we finish with the compilation of the problem definition, which is represented by a set of individuals. The problem description has two elements: the input description in terms of meta-data (characteristics of the data like attributes, types, median, etc.) and the goals/hints entered by the user. Both are stored as a set of ABox assertions. Having specified the planning domain and the problem description, one can start planning DM workflows.

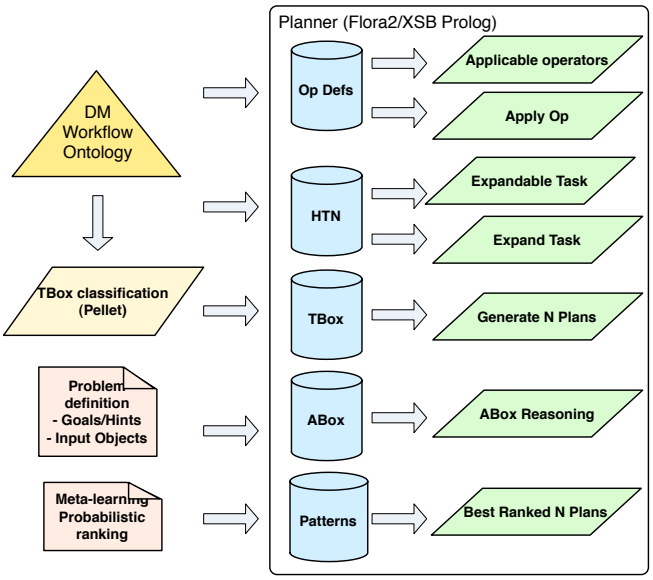


Figure 8: Workflow Ontology and AI Planner capabilities.

eIDA is a programming interface to the reasoner & planner used

[14] http://clarkparsia.com/pellet/
[15] http://www.w3.org/TR/owl2-profiles/

to plugin such an Intelligent Discovery Assistant (IDA), based on the services of the planner, into existing systems (so far into RapidMiner and Taverna).[16] It provides methods for retrieving the plans starting from the data set meta-data and the selection of a main goal. To improve the user experience with the RM-IDA plugin we have developed a simple installer based on precompiled binaries. It works on Linux, Mac OS X 10.5/6, Windows 7 and Windows XP systems.

The RapidMiner IDA Extension can be downloaded (or even auto-installed) from the Rapid-I Marketplace. [17] So far it was downloaded over 150 times during the first two months.

Both presented tools (eProPlan, eIDA) are open source and available on request.

An alternative implementation of RapidMiner IDA Extension exists for Taverna[18]. Taverna can execute all workflows composed of web-services. It can execute the workflows generated by the IDA[19] using any RapidAnalytics [20] server that provides all RapidMiner operators as web-services. Extensions for other KDD tools (e.g., KNIME, Enterprise Miner, etc.) would require two steps: first modeling their corresponding operators in the DMWF, second an implementation of the GUI and the plan-converter using the IDA-API.

4 Evaluation of the IDA

We tested the IDA on 108 datasets from the UCI repository of Machine Learning datasets. [21] It produced executable plans for all 78 classification and 30 regression problems. These datasets have between three and 1558 attributes, being all nominal (from binary too many different values like ZIP), all scalar (normalized or not), or mixed types. They have varying degrees of missing values. We are not aware of any other Machine Learning or DM approach that is able to adapt itself to so many different and divergent datasets. The IDA also works for less well prepared datasets like the KDD Cup 1998 challenge data (370 attributes, with up to 50% missing values and nominal data, where it generates plans of around 40 operators. Generating and ranking 20 of these workflows took 400 sec. on a 3.2 GHz Quad-Core Intel Xeon.

4.1 Ease of Use

Without an IDA data mining is typically achievable by specialized highly-trained professionals such as DM consultants. They have to know a lot about DM methods and how they are implemented in tools. They have to inspect the data and combine the operators into an adequate workflow.

The IDA reduces the technical burden, it now offers "DM with 7 clicks" (see Figure 5). (1) Show the IDA-Perspective of the tool; (2) drag the data to be analyzed from the repository to the view or import (and annotate) your data; (3) select your main goal in DM; (4) ask the IDA to generate workflows for data and goal; (5) evaluate all plans by executing them in RapidMiner; (6) select the plan you like most to see a summary of the plan (the screenshot in Figure 6 is made after this step); and finally, (7) inspect the plan and its results. Note that these steps do not require detailed technical knowledge anymore. Still a user should be aware of what (s)he is doing when (s)he uses

[16] http://www.taverna.org.uk/
[17] http://rapidupdate.de:8180/UpdateServer/faces/product_details.xhtml?productId=rmx_ida
[18] http://e-lico.eu/taverna-ida.html
[19] http://e-lico.eu/taverna-rm.html
[20] http://rapid-i.com/content/view/182/196/
[21] http://archive.ics.uci.edu/ml/

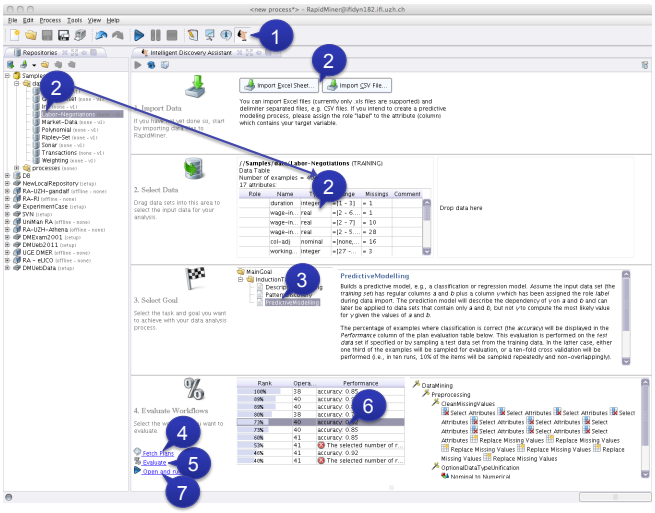


Figure 9: IDA Interface in RapidMiner

DM, i.e. (s)he should know the statistical assumptions underlying DM (e.g., a user should know what it means to have a sample that is representative, relevant, and large enough to solve a problem with DM/statistics). But this is knowledge required in any experimental science.

4.2 Speedup of Workflow Design

Besides making DM easier for inexperienced users, our main goal in building the IDA was to speed-up the design of DM workflows. To establish a possible speed-up we compared the efficiency of computer science students after attending a DM class to a person using the IDA. The study comprises in total 24 students (9 in 2011 and 15 in 2012). They had to solve the following DM problems:

- Take the UCI "Communities and Crime" data from `http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime` and
  - **a)** generate a fine clustering of data that allows me to look for very similar communities
  - **b)** generate a description of the clusters (learn a model to predict the cluster label build in task a)).
  - **c)** generate a function to predict "ViolentCrimesPerPop" and evaluate it with 10-fold cross-validation.
- Take the UCI "Internet Advertisement" data from `http://archive.ics.uci.edu/ml/datasets/Internet+Advertisements` and generate an evaluated classification for the attribute "Ad/Non-Ad".

All data are provided as already imported into RapidMiner (a local RapidAnalytics server they could access). All students took/needed the full 3 hours.

The study confirmed that the standard DM problems they had to solve (clustering and prediction tasks on complex UCI data) can be sped-up by the using an IDA whilst maintaing comparable quality: it took the students 3 hours (designing and executing the workflows) to solve the tasks, whereas a non-specialist using the IDA accomplished the same tasks in 30 minutes (IDA planning and minimal manual adaptation and execution of the workflows) with a comparable output. Table 10 shows how many students managed to solve successfully the given problems and how the IDA solved it.

Task	#Students Succeeded	#Students partial success	IDA's solution
Crime Data			
Clean Missing Values	22/24	1/24	drop & fill
Many Valued Nominals	11/24	6/24	drop
Normalization	-	-	yes
Clustering	9/24	-	2/10-means
Cluster Description	8/24	-	DT
Regression	RMSE$<$0.2:11/24, 0.22$\leq$RMSE$\leq$0.244: 3/11, Best RMSE=0.136 $\pm$0.010 : 4/24	-	k-NN RMSE=0.2
Evaluation	15/24	2/24	10-fold X-Val
Advertisement Data			
Clean Missing Values	17/24	1/24	fill
Feature Selection	no	no	no
Classification	acc$>$96%:13/24, 86%$\leq$acc$<$96%:6/24, Best Acc=97.32%:1/24	-	DT, Acc=96.34% $\pm$ 0.65
Evaluation	5/24	7/24	10-fold X-Val

Figure 10: Features of the designed solutions by students and IDA

Both datasets had missing values which could be ignored (−), the attribute could be all dropped or all filled or depending on the amount of missing values individually dropped & filled. Here, both students and IDA did a good job. The "Communities and Crime" data had key-like attributes (many valued nominals) which are likely to disturb the DM results and should be dropped or marked as (to be) ignore(d). Here, only around half of the students handled it correctly. Numerical attributes with different scales cause unwanted weighting of attributes in distance based similarity computation. Therefore, they should be normalized[22]. There are several clustering methods in RapidMiner. The best way to solve the clustering task is by using hierarchical top-down 2-means (k-means with $k = 2$) clustering till the grouping is fine enough. Only one student used this approach. The rest of the students and the IDA used k-means with different values for k (k$<$20 is successful, larger values make the prediction very difficult).The IDA sets $k = 2$ and there is no way to specify the goal of a "fine clustering". This can be solved by opening the returned workflow and changing the parameter (re-running it is not much effort). We manually choose $k = 10$ as we had the next task in mind and knew it is difficult to predict a nominal with too many different values[23], but many students chose a too fine k and failed the cluster description task (using different methods like Naive Bayes (NB), Decision Tree (DT) or jRIP), most only build bad models using the not dropped key-like attributes). For the "ViolentCrimesPerPop", most students used linear regression (linReg). The probabilistic ranking of the IDA preferred k-nearest neighbor. Common mistakes for this task were: regression applied on the wrong attribute, converting numeric data to nominal and applying NaiveBayes (bad accuracy), or converting it to binary: no crime (3 examples), crime (595 examples) the resulting model of course predicted crime everywhere. The DM step should have used a 10-fold cross validation, but some students delivered a training/test set split (maybe to save execution time, maybe because that was used in the group exercise). "Internet Advertisement" data has many attributes, so Feature Selection would have been an option, but no one did it, also the IDA had none in the top 5 ranked plans. The task was a simple binary prediction, non-ads are much more frequent (2820 non-ad, 459 ad), solved by all students by different methods. One balanced the data, but that worsened the results. One learned a decision tree but did not do an evaluation of the results. Some students even failed to produce a model or plot the data incorrectly (20:80).

This user evaluation provides a strong indication about the strength of the IDA. Note that the students are an optimal user-group for the IDA, as they have limited DM experience but understand the principles of DM.

[22] In fact the initial data is 0-1 range normalized, so the students did not do that step, but preprocessing operations like filling missing values change the column statistics. The planner cannot predict the results very well for column groups, so it ensures normalization of the data at the end of preprocessing.

[23] The meta-data analysis returns categorial for ≤ 10 different values and nominal otherwise. Prediction requires a categorial target (or numeric) target, i.e. the IDA refuses to build a plan to predict a target with more than 10 nominal values.

4.3 Performance of the generated workflows

The performance of the generated workflows depends strongly on the ranking. The baseline strategy is to rank the workflows simply based on the popularity of the operators. RapidMiner automatically collects operator usage-frequencies from all the users who accept to submit it. A workflow is ranked *better*, if it contains more frequently used operators. This already produces workflows comparable to user-designed workflows and was used in the speedup-experiments. Our e-LICO project partners[24] used the planner to systematically generate workflows, executed them to get the performance data, and applied meta-learning to these experiments [11].

In [10] they evaluated the meta-mining module and the resulting plan ranking on 65 biological datasets. These datasets are high dimensional with few instances/samples. For their experiments they cross-validated all performance by holding out a dataset. The resulting meta-model was then used to rank the IDA-generated workflows. They found that the meta-learned rankings significantly outperformed the default, frequency-based strategy. Hence, their ranker was able to improve on our ranking to find DM workflows that maximize predictive performance.

5 Conclusion

We presented our Intelligent Discovery Assistant (eIDA and eProPlan) for planning KDD workflows. eIDA can be easily integrated into existing DM-suites or workflow engines. eProPlan is a user-friendly environment for modeling DM operators and defining the HTN grammar for guiding the planning process. Furthermore, it is able to plan attribute-wise operations. The main scientific contribution of the IDA is the ability to build complex workflows out of a much larger set of operations than all previous systems. The demo presents how planning-based KDD workflow design can significantly help KDD practitioners to make their daily work more efficient.

ACKNOWLEDGEMENTS

The work described in this paper is partially supported by the European Community 7^{th} framework program ICT-2007.4.4 under grant number 231519 "e-Lico: An e-Laboratory for Interdisciplinary Collaborative Research in DM and Data-Intensive Science" and the result of collaborative work within this project. We especially like to thank Phong Nguyen, Alexandros Kalousis, Melanie Hilario, and Tom Smuc for their work on meta-learning and ranking, sketched here to show the overall picture.

[24] http://www.e-LICO.eu/

REFERENCES

[1] A. Bernstein, F. Provost, and S. Hill, 'Towards Intelligent Assistance for a Data Mining Process: An Ontology-based Approach for Cost-sensitive Classification', *IEEE Transactions on Knowledge and Data Engineering*, **17**(4), 503–518, (April 2005).

[2] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, 'Crisp–dm 1.0: Step-by-step data mining guide', Technical report, The CRISP–DM Consortium, (2000).

[3] C. Diamantini, D. Potena, and E. Storti, 'KDDONTO: An Ontology for Discovery and Composition of KDD Algorithms', in *Proceedings of the SoKD-09 Workshop at ECML/PKDD*, (2009).

[4] J. Gama and P. Brazdil, 'Characterization of classification algorithms', *Progress in Artificial Intelligence*, 189–200, (1995).

[5] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory & Practice*, Morgan Kaufmann, San Francisco, CA, USA, 2004.

[6] N. Jankowski and K. Grabczewski, 'Building meta-learning algorithms basing on search controlled by machine complexity', in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008*, pp. 3601–3608. IEEE, (2008).

[7] A. Kalousis and M. Hilario, 'Model selection via meta-learning: a comparative study', in *International Journal on Artificial Intelligence Tools, ICTAI 2000*, pp. 406–413. IEEE, (2000).

[8] J.-U. Kietz, F. Serban, A. Bernstein, and S. Fischer, 'Towards cooperative planning of data mining workflows', in *Proceedings of the SoKD-09 Workshop at ECML/PKDD*, pp. 1–12, (2009).

[9] J.-U. Kietz, F. Serban, A. Bernstein, and S. Fischer, 'Data mining workflow templates for intelligent discovery assistance and auto-experimentation', in *Proceedings of the SoKD-10 Workshop at ECML/PKDD*, (2010).

[10] P. Nguyen and A. Kalousis, 'Evaluation report on meta-miner'. Deliverable 7.2 of the EU-Project e-LICO, January 2012.

[11] P. Nguyen, A. Kalousis, and M. Hilario, 'A meta-mining infrastructure to support kd workflow optimization', in *Proc. of the PlanSoKD-11 Workshop at ECML/PKDD*, (2011).

[12] L. Rendell, R. Seshu, and D. Tcheng, 'Layered concept learning and dynamically-variable bias management', in *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI-87*, pp. 308–314. Citeseer, (1987).

[13] F. Serban, J. Vanschoren, J.-U. Kietz, and A. Bernstein, 'A Survey of Intelligent Assistants for Data Analysis', *ACM Computing Surveys*, (to appear 2012).

[14] D. Tsarkov and I. Horrocks, 'FaCT++ description logic reasoner: System description', *Lecture Notes in Computer Science*, **4130**, 292, (2006).

[15] R. Wirth, C. Shearer, U. Grimmer, T. P. Reinartz, J. Schlösser, C. Breitner, R. Engels, and G. Lindner, 'Towards process-oriented tool support for knowledge discovery in databases', in *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery, PKDD '97*, pp. 243–253, London, UK, (1997). Springer-Verlag.

[16] G. Yang, M. Kifer, and C. Zhao, 'Flora-2: A Rule-Based Knowledge Representation and Inference Infrastructure for the Semantic Web', *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, 671–688, (2003).

[17] M. Žaková, P. Křemen, F. Železný, and N. Lavrač, 'Automating knowledge discovery workflow composition through ontology-based planning', *IEEE Transactions on Automation Science and Engineering*, **8**(2), 253 –264, (2011).

[18] M. Žáková, V. Podpečan, F. Železný, and N. Lavrač, 'Advancing data mining workflow construction: A framework and cases using the orange toolkit', in *Proceedings of the SoKD-09 Workshop at ECML/PKDD*, (2009).

[19] M. Znidarsic, M. Bohanec, N. Trdin, T. Smuc, M. Piskorec, and M. Bosnjak, 'Non-functional workflow assessment'. Deliverable D7.3 of the EU Project e-LICO, November 2011.